

TDTS06

Lab 1 report

Mayeul Gasc (mayga330@student.liu.se)

Pierre Marsaa (piema262@student.liu.se)

Task A

- 1) We can read in the GET method in the first packet '*Request Version: HTTP/1.1*' we deduce that the HTTP version of the client is 1.1. The following package gives us the version used by the server '*Response Version: HTTP/1.1*', also version 1.1.
- 2) The first GET request from the client gives us several information about the client's browser. First, the languages accepted by the browser '*Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.5,en;q=0.3*' which means that the desired language is France french, or failing that Basic french, US english or english (in this order of priority). The packet also gives information about the accepted compression methods, the browser used, the operating system and accepted type of data (html, xml, etc.).
- 3) The client being the one using the GET method, we deduce from the first packet that the address of the client (the source) is 155.4.150.119 and that of the server (the destination) is 128.119.245.12.
- 4) The returned status code is 200, which means that the request was successful.
- 5) The last modification on the page is indicated as having taken place approximately 7 minutes before the request.
- 6) The packet contains the number of content bytes returned by the server : 128 bytes.
- 7) No, there isn't.

In this first part, we saw how a browser obtains a web page with the HTTP protocol. A GET request is sent with all the necessary information to the server to provide an appropriate response. We also notice that if we try to load the page again, the response from the server changes and indicates that there has been no modification and that the browser can try to load the cache (code 304). This is probably done to limit the consumption of bandwidth.

The image shows a Wireshark packet capture of an HTTP GET request and response. The packet list at the top shows two packets: a GET request (No. 127) and a 200 OK response (No. 131). Red arrows point to specific fields in the packet details pane with labels:

- IP addresses from the computer and the server.** Points to the Source and Destination IP addresses in the Ethernet II and Internet Protocol Version 4 sections.
- Status code returned from the server.** Points to the 200 OK status code in the Hypertext Transfer Protocol section.
- HTTP version.** Points to the HTTP/1.1 version in the Hypertext Transfer Protocol section.
- Other informations provided by the server.** Points to the User-Agent, Accept, and Accept-Language headers in the Hypertext Transfer Protocol section.
- Accepted languages.** Points to the Accept-Language header in the Hypertext Transfer Protocol section.

The packet details pane shows the following information for the selected packet (No. 131):

- Frame 127: 446 bytes on wire (3568 bits), 446 bytes captured (3568 bits) on interface \Device\NPF_{09400492-0031-4400-86C1-10EA4280095}, id 0
- Ethernet II, Src: Compalln_38:52:23 (98:29:a6:38:52:23), Dst: IETF-VNRP-VKID_1e (00:00:5e:00:01:1e)
- Internet Protocol Version 4, Src: 155.4.150.119, Dst: 128.119.245.12
- Transmission Control Protocol, Src Port: 56349, Dst Port: 80, Seq: 1, Ack: 1, Len: 392
- Hypertext Transfer Protocol
 - GET /wiresark-labs/HTTP-wiresark-file1.html HTTP/1.1
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 - Accept-Language: en-US,en;q=0.5
 - Accept-Encoding: gzip, deflate
 - Connection: keep-alive
 - Upgrade-Insecure-Requests: 1

No.	Time	Source	Destination	Protocol	Length	Info
127	5.464584	155.4.150.119	128.119.245.12	HTTP	446	GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
131	5.597551	128.119.245.12	155.4.150.119	HTTP	539	HTTP/1.1 200 OK (text/html)

Internet Protocol Version 4, Src: 128.119.245.12, Dst: 155.4.150.119

Transmission Control Protocol, Src Port: 80, Dst Port: 56349, Seq: 1, Ack: 393, Len: 485

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Date: Sun, 06 Sep 2020 08:35:52 GMT\r\n

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.9 mod_perl/2.0.11 Perl/v5.16.3\r\n

Last-Modified: Sun, 06 Sep 2020 05:59:02 GMT\r\n

Etag: "88-Sae9ecc79b22e"\r\n

Accept-Ranges: bytes\r\n

Content-Length: 128\r\n

Keep-Alive: timeout=5, max=100\r\n

Connection: Keep-Alive\r\n

Content-Type: text/html; charset=UTF-8\r\n

\r\n

[HTTP response 1/1]

[Time since request: 0.132967000 seconds]

[Request in frame: 127]

[Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]

File Data: 128 bytes

Line-based text data: text/html (4 lines)

Last modification

Bytes of content returned to the browser

Task B

- 8) No, there is no if-modified-since line.
- 9) Yes, the server clearly sent the entire page, because we see a 371 byte "data" field which corresponds to an html page.
- 10) The second request has the if-modified-since field with the date of the first packet.
- 11) While the first request gets the same response as in part A, the second gets a smaller response than the first time with the code 304 "not modified". We understand here that the browser asked the server if the page has been modified and that the latter answered no. The browser therefore loaded the cached page.

We observe in this part that the HTTP protocol is designed so as not to transfer unnecessary data. Indeed, the browser when it requests a page from the server, it will keep it in cache and in case the user requests the same page again, it specifies in the request that it already has a cached page. If no changes have taken place, the page is not resent and code 302 sent. The browser then displays the cached page.

No.	Time	Source	Destination	Protocol	Length	Info
8	3.532321	155.4.150.119	128.119.245.12	HTTP	446	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
12	3.670510	128.119.245.12	155.4.150.119	HTTP	783	HTTP/1.1 200 OK (text/html)
25	5.229800	155.4.150.119	128.119.245.12	HTTP	538	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
29	5.361207	128.119.245.12	155.4.150.119	HTTP	292	HTTP/1.1 304 Not Modified

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Date: Sun, 06 Sep 2020 09:22:05 GMT\r\n

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.9 mod_perl/2.0.11 Perl/v5.16.3\r\n

Last-Modified: Sun, 06 Sep 2020 05:59:02 GMT\r\n

Etag: "173-Sae9ecc79b22e"\r\n

Accept-Ranges: bytes\r\n

Content-Length: 371\r\n

Keep-Alive: timeout=5, max=100\r\n

Connection: Keep-Alive\r\n

Content-Type: text/html; charset=UTF-8\r\n

\r\n

[HTTP response 1/2]

[Time since request: 0.144189000 seconds]

[Request in frame: 8]

[Next request in frame: 25]

[Next response in frame: 29]

[Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]

File Data: 371 bytes

Line-based text data: text/html (10 lines)

HTTP status code and phrase

If-Modified-Since

Task C

- 12) The browser sends only one GET request, the number of the packet is 543.
- 13) The packet associated with the response is 559 with the code 200 (request OK).
- 14) It took four TCP packets to transfer the web page. (only packet containing the segmented data)
- 15) Since the HTTP and TCP protocols are not at the same layer, the HTTP is encapsulated in the TCP, so there is no communication between the two.

First and unique HTTP packet sent by the client

Server's response

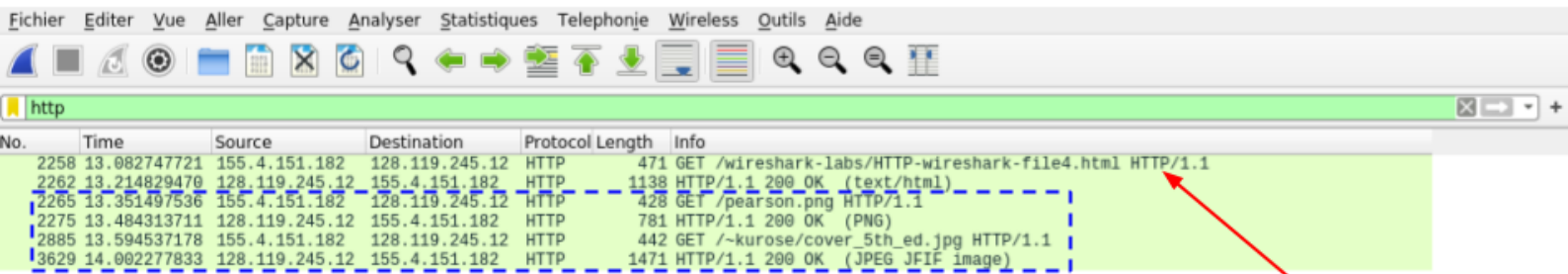
The four TCP packets sent by the server to carry the web page

In this part, we've seen that some HTTP responses are too big to fit in one packet. That's why the responses can be divided into a certain number of TCP packets. The number of TCP packets depend on the size of the HTTP response (1460 bytes per packet). Both protocols aren't on the same layer and therefore do not communicate with each other. It is only the TCP protocol that should be concerned with dividing the message into packets.

Task D

- 16) Three requests were sent by the browser, one for the HTML code of the page and two others for the two images. All requests are sent to the site gaia.cs.umass.edu (128.119.245.12).
- 17) The two requests were sent in parallel, we notice on the capture that the GET requests are grouped, as well as the responses to these requests.

With this task, we saw that each image needed one request for itself and that the request can be done in parallel (twice get, twice responses) or serially (one get, one response, one get, one response). During our various tests, we observed both methods.



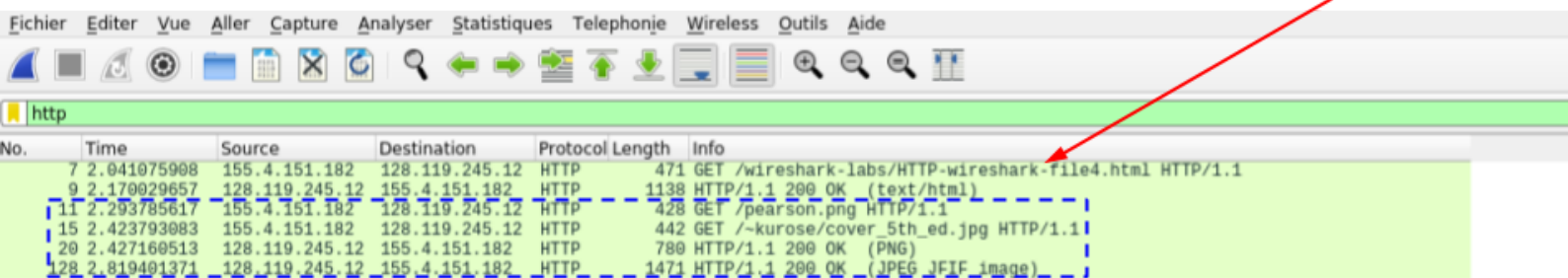
The screenshot shows a Wireshark capture of HTTP traffic. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
2258	13.082747721	155.4.151.182	128.119.245.12	HTTP	471	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
2262	13.214829470	128.119.245.12	155.4.151.182	HTTP	1138	HTTP/1.1 200 OK (text/html)
2265	13.351497536	155.4.151.182	128.119.245.12	HTTP	428	GET /pearson.png HTTP/1.1
2275	13.484313711	128.119.245.12	155.4.151.182	HTTP	781	HTTP/1.1 200 OK (PNG)
2885	13.594537178	155.4.151.182	128.119.245.12	HTTP	442	GET /~kurose/cover_5th_ed.jpg HTTP/1.1
3629	14.002277833	128.119.245.12	155.4.151.182	HTTP	1471	HTTP/1.1 200 OK (JPEG JFIF image)

Red arrows point from the text labels to specific rows in the table.

Serials GET images requests

GET request of the HTML code



The screenshot shows a Wireshark capture of HTTP traffic. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
7	2.041075908	155.4.151.182	128.119.245.12	HTTP	471	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
9	2.170029657	128.119.245.12	155.4.151.182	HTTP	1138	HTTP/1.1 200 OK (text/html)
11	2.293785617	155.4.151.182	128.119.245.12	HTTP	428	GET /pearson.png HTTP/1.1
15	2.423793083	155.4.151.182	128.119.245.12	HTTP	442	GET /~kurose/cover_5th_ed.jpg HTTP/1.1
20	2.427160513	128.119.245.12	155.4.151.182	HTTP	780	HTTP/1.1 200 OK (PNG)
28	2.819401371	128.119.245.12	155.4.151.182	HTTP	1471	HTTP/1.1 200 OK (JPEG JFIF image)

Red arrows point from the text labels to specific rows in the table.

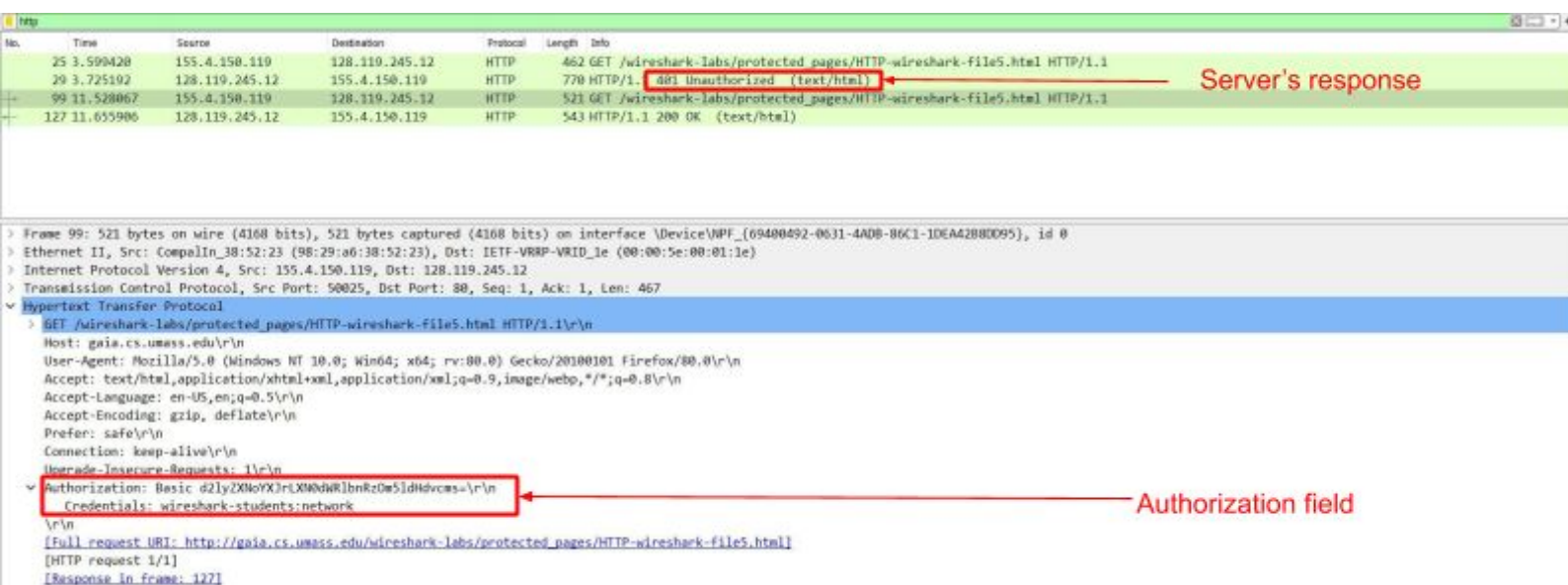
GET images requests in parallel

Task E

18) The server's first response was an access denied with the code 401.

19) In the second request, a new line was there: 'Authorization' (with the connection information entered)

Here we have an example of authentication via HTTP protocol. The client first asks to access a web page and the server responds the first time indicating access denied. The browser then invites the user to enter the identifiers that it sends to the server in a new request. If the



The screenshot shows a Wireshark capture of HTTP traffic. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
25	3.590420	155.4.150.110	128.119.245.12	HTTP	462	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
29	3.725102	128.119.245.12	155.4.150.110	HTTP	770	HTTP/1.1 401 Unauthorized (text/html)
99	11.520067	155.4.150.110	128.119.245.12	HTTP	521	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
127	11.655906	128.119.245.12	155.4.150.110	HTTP	543	HTTP/1.1 200 OK (text/html)

Below the packet list, the details of the selected packet (No. 99) are shown:

```
> Frame 99: 521 bytes on wire (4168 bits), 521 bytes captured (4168 bits) on interface \Device\NPF_{69400492-0631-4A0B-86C1-IDEA2B80095}, id 0
> Ethernet II, Src: Compalin_38:52:23 (98:29:a6:38:52:23), Dst: IETF-VRRP-VRID_1e (00:00:5e:00:01:1e)
> Internet Protocol Version 4, Src: 155.4.150.110, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 50025, Dst Port: 80, Seq: 1, Ack: 1, Len: 467
> Hypertext Transfer Protocol
  > GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: safe\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Authorization: Basic d2lyZXNoYXJrLnRlcXNDdWRJbnRzOW5ldHdvcmVz\r\n
    Credentials: wireshark-students:network\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html]
    [HTTP request 1/1]
    [Response in frame 127]
```

Red arrows point from the text labels to specific parts of the capture.

credentials are correct, the response from the server contains the desired web page. We notice that the identifiers are sent in clear and that there is a security problem.

HTTP Persistent connection

- 20) According to our research on the internet, these fields indicate whether the HTTP connection should be maintained or not. With version 1.1 of the HTTP protocol, the connection is not systematically closed after the end of the request so as not to have to open a new one if browsing continues on the same website.